70th International Astronautical Congress (IAC), Washington D.C., U.S.A., 21-25 October 2019.

IAC–19–C1.2.1

Multi-Objective Robust Trajectory Optimisation Under Epistemic Uncertainty and Imprecision

Simão Graça Marto[a]*, Massimiliano Vasile[a], Richard Epenoy[b]

[a]*Department of Mechanical & Aerospace Engineering,University of Strathclyde, James Weir Building, 75 Montrose Street, Glasgow, United Kingdom G11XJ*,
simao.da-graca-oliveira-marto@strath.ac.uk
[b]*Centre National d'Études Spatiales*, richard.epenoy@cnes.fr
*Corresponding Author

Abstract

This paper presents a novel method to generate robust optimal trajectories for spacecraft equipped with low-thrust propulsion under the effect of epistemic uncertainty. The uncertainties considered for this paper derive from a lack of knowledge on system's and launcher's parameters. This is a typical situation in the early stage of the design process when multiple options need to be evaluated and only a partial knowledge of each of them is available. Uncertainties are modelled with probability boxes, or p-boxes, embodying multiple families of distributions. Once the effect of uncertainty is propagated through the system one can calculate the Upper and Lower Expectations on the quantity of interest (for example the mass of propellant). The Lower Expectation defines the worst case effect of the uncertainty when uncertainty is expressed via a p-box. We also propose a method for its calculation, which requires solving an optimization problem. Once the low expectations on the quantities of interest are available, a novel efficient computational scheme is proposed to compute families of control laws that are robust against the effect of uncertainty. Robustness is here considered to be the ability to maximise the desired performance, under uncertainty, with a high probability of satisfying the constraints. The computational scheme proposed in this paper makes use of surrogate models of the Lower Expectations, to radically reduce the computational cost of the robust optimisation problem. This is combined with a dimensionality reduction technique, that allows one to construct surrogate models on low dimensional spaces, and an iterative refinement of the surrogate representation. The training points of the surrogate models are evaluated using FABLE (Fast Analytical Boundary value Low-thrust Estimator), an analytical tool for the fast design and optimisation of low-thrust trajectories. A memetic multi-objective optimisation algorithm, MACS (Multi Agent Collaborative Search), is then used to find the set of Pareto optimal control laws that maximise the Lower Expectation in the achievement of the desired values of objective function and constraints. The proposed approach is then applied to the design of a rendezvous mission to Apophis with a small spacecraft equipped with a low thrust engine.

Keywords: Epistemic uncertainty, Resilient satellite, Robust Optimization, Lower Expectation, Multi-Objective Optimization

Nomenclature

$\boldsymbol{\xi}$	Uncertain variable
E_l	Lower expectation
Ξ	Uncertainty space
n_ξ	Number of uncertain variables
u	Control law
w	Proxy variable for control law
E	Expectation

Acronyms

FABLE Fast Analytical Boundary value Low-thrust Estimator

MACS Multi Agent Collaborative Search (optimization algorithm)

pdf Probability Density Function

1. Introduction

Asteroids in the solar system are interesting targets for space exploration missions both for scientific reasons and for planetary defense. The high delta-v requirements, coupled with the small gravitational accelerations, make low thrust engines suited for the purpose of asteroid rendezvous. Trajectories using these engines can not be modelled as being composed of instantaneous impulses, as is done with their chemical counterparts. Instead, the trajectory is split into arcs during which the engine is on or off and pointing in a certain direction. The parameters that describe these trajectories are consolidated in an array u (Equation 5). The challenging task of propagating and optimizing these trajectories is performed by FABLE, described in Section 2.

The epistemic uncertainty in the system's and launcher's parameters, characteristic of the early stage of the design process, provides a challenge for finding a solution that guarantees mission success under this uncertainty. Such a solution is termed robust. We will consider as uncertain variables the engine parameters at various points in the trajectory, as well as the Earth escape velocity provided by the launcher. One approach is to characterize this uncertainty using p-boxes, that is, a space of possible probability distributions over the uncertain space. The design parameters are then optimized considering the probability distribution in this space that represents the worst case scenario. More specifically, the lower expectation E_l is optimized. The detailed definition and calculation of this quantity is the subject of Section 4.

For a certain stochastic control problem, we have multiple metrics y_i. These can be, for example, the propellant mass (m_p), or the distance to a certain target (Δr). For certain thresholds ν_i on these variables, we wish to find a control vector u that maximizes the lower expectations on each of the indicator functions $y_i(u) < \nu_i$:

$$\max_{u} \{\mathrm{E}_l(y_i(u) < \nu_i)\ \forall i\} \tag{1}$$

Another interpretation for $\mathrm{E}_l(y_i(u) < \nu_i)$ is that it represents the lowest probability of y_i being below ν_i that can be obtained with the family of distributions considered. Since we're doing a multi-objective optimization, it's perfectly reasonable to also optimize the thresholds, so we could have:

$$\min_{u,\nu} \{-\mathrm{E}_l(y_i(u) < \nu_i);\ \ \nu_i\ \forall i\} \tag{2}$$

As shown in Section 2, the control law u we use has 67 dimensions. This high dimensionality is a problem given the computational complexity of the calculation of E_l, combined with the multi-objective optimization. To tackle this, we use control mapping to reduce the number of dimensions. A variable w, with less dimensions than u, is used as a proxy for u using a control map U, such that our problem becomes:

$$\min_{w,\nu} \{-\mathrm{E}_l(y_i(U(w)) < \nu_i);\ \ \nu_i\ \forall i\} \tag{3}$$

Details on how that control mapping works are in Section 3.

In our optimization problem we have multiple objectives, which all should be optimized in order for the mission to be likely to succeed. Sometimes optimizing one objective might conflict with optimizing another, i.e., there is a trade-off between the objectives. One way to gain knowledge of this trade-off, to assist with the design process, is to use a multi-objective optimizer to produce a Pareto front. This consists of a set of all solutions that do not dominate each other. We use MACS as a multi-objective optimizer, as well as Krigging surrogate models to speed-up convergence. This process is described in Section 5.

Our work follows a similar framework to Di Carlo et al. [1], except our method scales non-exponentially with the number of uncertain variables, we perform multi-objective optimization, and use lower expectation instead of evidence theory, among other differences.

2. Propagation and Optimization of Low Thrust Trajectories

The low thrust trajectories considered in this paper consist of an ejection by a conventional launcher, followed by a number of alternating coast and thrust arcs with ion propulsion. The ejection is characterized by the departure time t_D, and the magnitude v_∞, azimuth γ and declination δ of the escape velocity relative to the Earth in a heliocentric reference frame. The i^{th} coast arc is characterized by its length in longitude $\Delta L_{OFF,i}$, i.e., the difference between the longitude at the end of the arc and at the beginning. The i^{th} thrust arc is characterized by its length $\Delta L_{ON,i}$, and by the azimuth α_i and declination β_i that the spacecraft engine is pointing towards. Variables that are required to calculate a trajectory, but which are not part of the control vector, are the engine thrust at $r = 1\mathrm{AU}$, T, and the specific impulse I_{sp}. It is also possible to consider as control variables the throttle in each arch τ_i, and the times of arrival at each of possibly multiple targets $t_{T,i}$, but we do not optimize these variables.

The metrics y_i are calculated with FABLE-propagator, which is how we'll refer to the propagator that is part of FA-

BLE [2]. This uses formulas similar to those described and derived in [3], to quickly propagate using the equinoctial orbital parameters. The one difference is that we're considering a solar powered low-thrust engine, so the thrust is proportional to the inverse square of the heliocentric distance r:

$$T(r) = T\frac{1\text{AU}^2}{r^2} \tag{4}$$

The locations of the targets at the times of rendezvous t_R are calculated using a keplerian propagator. FABLE-propagator calculates the orbital parameters of the spacecraft at the longitudes L_R of the targets at t_R, as an approximation for the closest approach.

2.1 Optimal Trajectory

All of the variables described above, used with FABLE-propagator, define a direct transcription for the problem of optimizing a low-thrust trajectory. With FABLE [2], the user can fix some and optimize the rest of these variables. The variables that are optimized are contained in vector u, defined as:

$$u = [\mathbf{\Delta L_{OFF}}\ \mathbf{\Delta L_{ON}}\ \boldsymbol{\alpha}\ \boldsymbol{\beta}\ \gamma\ \delta\ t_D] \tag{5}$$

This is similar to the previous version as defined in [1], with the addition of the departure time t_D as an optimisable variable. We are considering 16 thrust and coast arcs, so variables $\mathbf{\Delta L_{OFF}}$, $\mathbf{\Delta L_{ON}}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are each a vector with 16 elements, each element corresponding to each arc, for a total of 67 elements in u. It is common for arc lengths to be zero after optimization, corresponding to having a lower number of arcs in practice.

Some of the variables that are not optimized can be either fixed variables that are known a-priori, or they could be uncertain variables $\boldsymbol{\xi} \in \Xi \subset \mathbb{R}^{n_\xi}$. FABLE can only optimize for a deterministic setting, i.e., with set values for $\boldsymbol{\xi}$, but in our work FABLE-propagator was updated to be able to quickly propagate for large numbers of different $\boldsymbol{\xi}$ values in parallel, using CPU vectorization. This is used to quickly generate many samples of the metrics for a single control under different uncertain variables. These samples are then used with quasi-Monte Carlo methods to characterize the uncertainty in the mission objectives. This turned out to be faster than using surrogate models, as was done in [1]. The usage of these samples to characterize the uncertainty, by obtaining E_l, is discussed in more detail in section 4. As an example, the uncertain variables could be:

$$\boldsymbol{\xi} = [v_\infty\ T\ I_{sp}] \tag{6}$$

FABLE uses MATLAB's fmincon to find the solution to Program 7, where the propellant mass m_p is minimized, with the constraint that the spacecraft flies by or rendezvous with each target. If a target is marked as fly-by, the constraint is that the position of the spacecraft at L_R matches that of the spacecraft, and if the target is marked as rendezvous, both position and velocity have to match. It is also necessary to guarantee that the spacecraft reaches this point at time t_R.

$$\begin{aligned} &\min_u m_p(u, \boldsymbol{\xi}) \\ s.t. \forall i \quad &r_{f,i}(u, \boldsymbol{\xi}) = r_{T,i} \\ &v_{f,i}(u, \boldsymbol{\xi}) = v_{T,i} \text{ if } i \text{ is not fly-by only} \\ &t_{f,i}(u, \boldsymbol{\xi}) = t_{T,i} \end{aligned} \tag{7}$$

Where each i represents each target. Also, $r_{f,i}$, $v_{f,i}$ and $t_{f,i}$ indicate the position, velocity and time of the spacecraft when its longitude matches that of the target i at time $t_{T,i}$. When a target is not fly-by only, the constraint that is imposed is that all equinoctial elements match, which is equivalent to requiring that the position and velocity both match.

FABLE-propagator is a tool that propagates trajectories with low-thrust engines. Its use in our work is to provide direct transcription to FABLE, and to perform Monte-Carlo analysis of the uncertainty in mission objectives.

FABLE is a tool designed to find optimal trajectories in a deterministic scenario. Its use in our work is as a way to reduce the dimensionality of the stochastic optimization problem via control mapping, as described in the next Section.

3. Control Mapping

We wish to find the control law that minimizes the lower expectation. The control law as defined in Equation 5, with 16 thrust and coast arcs, has 67 dimensions. Optimizing a nonlinear nonconvex function over such a high dimensional space would be very expensive. Instead, a control mapping strategy is used, which reduces the dimensionality of the problem, making it more suitable for use with global optimiser and surrogate models.

Intuitively, we should expect that a control vector that optimizes the stochastic problem should also be suitable for the deterministic problem where we fix the uncertain variables to some value $\boldsymbol{\xi}_u \in \Xi$. With this intuition we formulate a control map $U : \boldsymbol{\xi}_u \longrightarrow u$, where we solve Program 7

with $\boldsymbol{\xi} = \boldsymbol{\xi}_u$. This way $\boldsymbol{\xi}_u$ is a proxy variable for the control law. The stochastic program becomes Equation 3.

3.1 Reachable Set Mapping

In some cases we found that the above control map was too restrictive. The space of control vectors can be increased by aiming for a final position and velocity which is near, but not exactly equal, to the target's. The displacements in both these quantities are D_r and D_v. Instead of using $\boldsymbol{\xi}_u$ as a proxy for the control vector, we use $w = (\boldsymbol{\xi}_u, D_r, D_v)$, and instead of the control map being defined by equation 7, it's defined as:

$$\begin{aligned} & \min_u m_p(u, \boldsymbol{\xi}_u) \\ s.t. \forall i \quad & r_{f,i}(u, \boldsymbol{\xi}_u) = r_{T,i} + D_r \\ & v_{f,i}(u, \boldsymbol{\xi}_u) = v_{T,i} + D_v \text{ if } i \text{ is not fly-by only} \\ & t_{f,i}(u, \boldsymbol{\xi}_u) = t_{T,i} \end{aligned} \tag{8}$$

This increases the dimensionality of our problem, enough to increase the space of solutions that we can search so as to improve our end result, but not too much that it would make the problem take too long to solve.

4. Epistemic Uncertainty

When there is epistemic uncertainty, we do not know which distribution our uncertain variables $\boldsymbol{\xi}$ follow. Multiple conflicting sources of information may suggest different distributions, belonging to a family. Therefore, the expectation of some variable can be any value within an interval bounded by the lower and the upper expectations.

We obtain robust solutions by optimizing the lower expectations (E_l) of functions of interest I, which in our case will be defined as indicator functions $I = y < \nu$, representing whether a metric y is below a certain threshold ν. The lower expectation E_l is defined as the minimum expectation of the function of interest with respect to a family of probability distributions $\mathcal{P}$:

$$\mathrm{E}_l(I) = \min_{p \in \mathcal{P}(\Xi)} \int_\Xi I(\boldsymbol{\xi}) p(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi} \tag{9}$$

We follow [4] and define the family of probability distributions $\mathcal{P}$ using Bernstein polynomials. These Bezier curves are a linear combination of positive basis functions. If the coefficients are positive, the resulting function is guaranteed to be positive, which makes them ideal to represent probability distributions.

We could define the family of distributions as a linear combination of multi-variate Bernstein basis functions, as in Equation 10. This allows approximating any multi-variate distribution (see for example Section 7.4 of [5]), including those of correlated variables. The problem of finding the lower expectation is a linear program, but the complexity is exponential with n_ξ, as the number of coefficients is $(q+1)^{n_\xi}$, q being the order of the polynomial.

$$\mathcal{P}_m = \left\{ \begin{array}{l} \displaystyle\sum_{\mathbf{j} \in \mathcal{J}} c_{\mathbf{j}} B_{\mathbf{j}} \left(\tau(\boldsymbol{\xi}) \right) \\ \forall \mathbf{c} > 0 : \displaystyle\sum_{\mathbf{j} \in \mathcal{J}} c_{\mathbf{j}} = 1 \end{array} \right\} \tag{10}$$

where $\mathbf{j} = \{j_1, \ldots, j_{n_\xi}\}$ represents a tuple of n_ξ indexes and $\mathcal{J} = \{0, \ldots, q_1\} \times \ldots \times \{0, \ldots, q_{n_\xi}\} \subset \mathbb{N}^{n_\xi}$ represents the set of possible such tuples.

An alternative is having the distributions be the product of univariate Bernstein polynomials, as in equation 11. In this new family of distributions only independent variables are possible, since the distributions are defined as the product of univariate functions. The number of coefficients is now $(q+1) \times n_\xi$, which no longer grows exponentially with the number of uncertain variables. However, Problem 9 is no longer a linear program.

$$\mathcal{P}_u = \left\{ \begin{array}{l} p(\boldsymbol{\xi}; \mathbf{c}) = \displaystyle\prod_{k=1}^{n_\xi} \sum_{j=0}^{q_k} c_j^{(k)} b_{j;q_k}(\tau_k(\xi_k)) \\ \forall \mathbf{c} > 0 : \displaystyle\sum_j c_j^{(k)} = 1 \; \forall k \end{array} \right\} \tag{11}$$

In both of these formulas, τ is a linear function such that $\tau(\xi) \in [0 \; 1]^{n_\xi} \; \forall \xi \in \Xi$. Also, b and B are Bernstein basis functions scaled so that they're both pdfs, that is, they integrate to 1 in Ξ. Their definitions, therefore, are:

$$b_{j;q}(x) = (q+1) \binom{q}{j} x^j (1-x)^{q-j} \tag{12}$$

$$B_{i_1,\ldots,i_{n_\xi};q_1,\ldots,q_{n_\xi}}(\mathbf{x}) = \prod_{k=1}^{n_\xi} b_{j_k;q_k}(x_k) \tag{13}$$

where q indicates the degree of a polynomial.

We want an algorithm that does not grow exponentially with n_ξ, so we will be considering family $\mathcal{P}_u$ to define the lower expectation. We will see in Subsection 4.2 that either of the families of distributions would produce the same lower expectation with Equation 9.

We define $\mathrm{E}(I;\mathbf{c})$ as the expectation of I obtained by using the distribution $p(\boldsymbol{\xi};\mathbf{c})$ corresponding to using $\mathbf{c}$ in Equation 11:

$$\mathrm{E}(I;\mathbf{c}) = \int_{\Xi} I(\boldsymbol{\xi}) p(\boldsymbol{\xi};\mathbf{c}) \mathrm{d}\boldsymbol{\xi} \tag{14}$$

4.1 Estimating Expectation

Given the difficulty of calculating the integral in Equation 14 analytically, a quasi-Monte Carlo approach is employed. This is the same as a Monte Carlo approach, but instead of selecting the samples randomly, a deterministic, low-discrepancy sequence $\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N)}$ is used. We use the Halton sequence for this purpose.

At this point, one way to estimate expectation $\mathrm{E}(I;\mathbf{c})$ would be the following:

$$\hat{E}_U(I;\mathbf{c}) = \frac{1}{N}\sum_{i=1}^{N} I(\boldsymbol{\xi}^{(i)}) p(\boldsymbol{\xi}^{(i)};\mathbf{c}) \,, \quad \boldsymbol{\xi}^{(i)} \sim U \tag{15}$$

Here, $\boldsymbol{\xi}$ is not a random variable, but a deterministic sequence, the Halton sequence. The expression $\boldsymbol{\xi} \sim U$ means that the histogram of $\boldsymbol{\xi}$ approximates the pdf of distribution U, the uniform distribution. This corresponds to taking the Halton sequence as is, with no transformation. Later we will consider modifying the Halton sequence so that its histogram is not uniform.

The Koksma-Hlawka inequality [6] provides an upper bound on the absolute error in Equation 15, but as the number of dimensions increases, the upper bounds on discrepancy that can be found in the literature become useless, requiring huge ($> 10^{19}$ for $n_\xi = 10$) numbers of samples to become lower than one, which the discrepancy, by definition, has to be always.

Therefore, the analysis on the error was performed using probability theory, as if $\boldsymbol{\xi}$ were randomly sampled, and afterwards an experimental analysis was performed for a special case of indicator functions that can be easily integrated analytically.

Using probability theory, the standard deviation (σ) of Equation 15 can be written as

$$\sigma_U^2 = \frac{1}{N}\left(\int I^2 p^2 \mathrm{d}\boldsymbol{\xi} - \left(\int I p \mathrm{d}\boldsymbol{\xi}\right)^2\right) \leq \frac{1}{N}\int p^2 \mathrm{d}\boldsymbol{\xi}, \tag{16}$$

since $0 \leq I \leq 1$ in all domain. This expression can be calculated, given the separability of the pdf,

$$\sigma_U^2 \leq \frac{1}{N}\prod_{k=1}^{n_\xi}\int \left(\mathcal{B}_j(\xi_k)\right)^2 \mathrm{d}\xi_k \tag{17}$$

When all the weight in c is applied to the $j = 0$ or $j = q$, q being the degree of the polynomials in each dimension, that upper bound becomes:

$$\frac{1}{N}\left(\frac{(q+1)^2}{2q+1}\right)^{n_\xi} \tag{18}$$

which appears to be the highest value the variance can have, based on empirical results. A weight $\mathbf{c}$ of this type is often the minimizer of the Expectation in our case. This upper bound for the standard deviation grows exponentially with n_ξ. For example, if $n_\xi = 10$, $q = 4$ and we want the standard deviation to be below 0.01, we would need $N > 2.7 \times 10^8$ samples.

To reduce the number of samples required to obtain sufficient accuracy, we can apply importance sampling. Importance sampling is usually applied to Monte-Carlo estimations, but nothing logically prevents its application to quasi-Monte Carlo. It works because by changing the sampling of the variables and correcting the values of the samples appropriately, one can obtain an estimator that is also unbiased, but which has a different, hopefully lower, variance. In our case we can have $\boldsymbol{\xi} \sim P(\mathbf{c})$, henceforth referred to as "P-sampling", where $P(\mathbf{c})$ represents the pdf obtained by using $\mathbf{c}$ in Equation 11.

We obtain a sequence approximating a univariate distribution P with cdf $C(x)$, if we have $x_u \sim U$, by doing $x_P = C^{-1}(x_u) \sim P$. When we have a multivariate distribution of independent variables, such as $P(\mathbf{c})$, if $C_k(\mathbf{c})$ is the cdf along the k^{th} variable, we can do $\xi_{k,P} = C_k^{-1}(\xi_{k,U})$ to obtain a sampling $\boldsymbol{\xi}_P \sim P(\mathbf{c})$. This does restrict us to having $P(\mathbf{c})$ be a pdf of independent variables, but since we are using the family $\mathcal{P}_u$, this is not an issue. The inverse of C_k cannot, in general, be found analytically, but very good approximations can be obtained using linear interpolation, which is what we do.

With P-sampling, the estimator becomes:

$$\hat{E}_P(I;\mathbf{c}) = \frac{1}{N}\sum_{i=1}^{N} I(\xi_i) \,, \quad \xi_i \sim P(\mathbf{c}) \tag{19}$$

The standard deviation now follows

$$\sigma_P^2 = \frac{1}{N}\left(\int I(\xi)^2 p(\xi)\mathrm{d}\xi - E^2\right) \leq \frac{1}{N}(E - E^2) \leq \frac{1}{4N}\,, \tag{20}$$

again using the fact that $0 \leq I \leq 1$. This new expression does not increase with n_ξ, solving the exponential problem we had before. Now, to obtain a similar limit on the standard deviation as before, we only need 2500 samples. We use 5000 in this document.

The fact that we now need to re-sample each time we want to evaluate $\mathrm{E}(I;\mathbf{c})$ with a different $\mathbf{c}$ is a disadvantage, but, as we will see, the reduction in the number of samples evaluated for each $\mathbf{c}$ compensates for the fact that they have to be recalculated for each $\mathbf{c}$. Furthermore, the number of samples required to have standard deviation below 0.01 with uniform sampling and $n_\xi = 10$, $q = 4$, exceeds the RAM capacity. This means that the samples would either need to be stored in hard drive or recalculated each time anyway, both of which are slower than recalculating 5000 samples.

To confirm the advantage of using P-sampling, a numerical experiment was run. For a number of randomly generated hyper-rectangular indicator functions, the exact integral was calculated using analytical formulas, and estimated with both strategies considered here. Table 1 contains the maximum absolute errors with 10000 test cases obtained by comparing both estimations with the analytical result. In the first column we used uniform sampling with $N = 1.5 \times 10^6$ samples, and in the second we used P sampling with $N = 5000$ samples. The difference in the number of samples is to compensate for the fact that with P-sampling these samples will have to be calculated each time a new value of $\mathbf{c}$ is considered, which is unnecessary when uniform sampling was used.

If less than 300 different values of $\mathbf{c}$ are considered, doing P-sampling with $N = 5000$ samples will be less computationally demanding than uniform sampling with $N = 1.5 \times 10^6$. As shown in Figure 1, the algorithm evaluates less than 300 (100 per metric) different $\mathbf{c}$ before finding a local optimum. Looking at Table 1 we see that for $n_\xi \geq 5$, the error was smaller with P-sampling with $N = 5000$ samples, and therefore, it is preferable to use this method in those cases.

Table 1: Maximum absolute errors for different sampling techniques

	Uniform sampling $N = 1.5 \times 10^6$	P sampling $N = 5000$
n_ξ	max abs error	max abs error
1	8.3×10^{-6}	6.4×10^{-4}
2	5.5×10^{-5}	1.4×10^{-3}
3	3.1×10^{-4}	1.5×10^{-3}
4	1.0×10^{-3}	2.3×10^{-3}
5	3.1×10^{-3}	2.5×10^{-3}
6	1.1×10^{-2}	4.1×10^{-3}
7	2.9×10^{-2}	3.6×10^{-3}
8	7.9×10^{-2}	4.7×10^{-3}
9	1.7×10^{-1}	4.3×10^{-3}
10	5.2×10^{-1}	5.7×10^{-3}

4.2 Minimizing Expectation

To calculate the lower expectation we want to find $\mathbf{c}$ that minimizes the expectation:

$$\mathrm{E}_l(I) = \min_{\mathbf{c}} \mathrm{E}(I;\mathbf{c}) \tag{21}$$

The optimization variable $\mathbf{c}$ has $n_\xi \times (q+1)$ dimensions if $\mathcal{P} = \mathcal{P}_u$, and the problem we're optimizing is non-linear. It can be shown, however, to be equivalent to a non-linear integer problem with just n_ξ dimensions.

Lemma 1. *If E_l is defined using the family $\mathcal{P}_m$, there is always a vector $\hat{\mathbf{c}}$ that only has one entry equal to 1, and all others to zero, such that $\mathrm{E}_l(I) = \mathrm{E}(I, \hat{\mathbf{c}})$.*

Proof. The problem of finding E_l with $\mathcal{P} = \mathcal{P}_m$ can be written as

$$\min_{\mathbf{c}} \sum_{\mathbf{j}\in\mathcal{J}} \lambda_{\mathbf{j}} c_{\mathbf{j}} \tag{22}$$

where

$$\lambda_{\mathbf{j}} = \int_\Xi I(\boldsymbol{\xi}) B_{\mathbf{j}}(\tau(\boldsymbol{\xi}))\mathrm{d}\boldsymbol{\xi} \tag{23}$$

This is a linear problem. Let $\tilde{\mathbf{j}}$ be such that $\lambda_{\tilde{\mathbf{j}}} = \min_{\mathbf{j}} \lambda_{\mathbf{j}}$. We can write our original problem as:

$$\min_{\mathbf{c}} \lambda_{\tilde{\mathbf{j}}} + \sum_{\mathbf{j}\in\mathcal{J}} (\lambda_j - \lambda_{\tilde{\mathbf{j}}}) c_{\mathbf{j}} \geq \lambda_{\tilde{\mathbf{j}}} \tag{24}$$

It's now evident that $\min_{\mathbf{c}} \sum_{\mathbf{j}\in\mathcal{J}} \lambda_{\mathbf{j}} c_{\mathbf{j}} = \lambda_{\tilde{\mathbf{j}}}$, which can be obtained by making $c_{\tilde{\mathbf{j}}} = 1$ and $c_{\mathbf{j}} = 0\ \forall \mathbf{j} \neq \tilde{\mathbf{j}}$.

□

Lemma 2. *If E_l is defined using the family $\mathcal{P}_u$, there is always a coefficient vector $\hat{\mathbf{c}}$ such that the lower expectation verifies $\mathrm{E}_l(I) = \mathrm{E}(I, \hat{\mathbf{c}})$ with $\hat{\mathbf{c}}$ such that:*

$$\forall k \exists \tilde{j} : \hat{\mathbf{c}}_{\tilde{j}}^{(k)} = 1 \text{ and } \hat{\mathbf{c}}_j^{(k)} = 0\ \forall j \neq \tilde{j} \tag{25}$$

Proof. We will show that if we have a vector $\tilde{\mathbf{c}}$ such that $\mathrm{E}_l(I) = \mathrm{E}(I, \tilde{\mathbf{c}})$, but which does not satisfy the constraint in the lemma, we can use it to obtain a vector $\hat{\mathbf{c}}$ that does.

If $\tilde{\mathbf{c}}$ does not satisfy that constraint, then for at least one s, we must have $\tilde{\mathbf{c}}^{(s)}$ with more than one non-zero element. Let's consider the problem of optimizing $\min_{\mathbf{c}^{(s)}} \mathrm{E}(I, \mathbf{c})$, where we fix all elements of $\mathbf{c}$ except for those in $\mathbf{c}^{(s)}$. This can be written as:

$$\min_{\mathbf{c}^{(s)}} \sum_{j=0}^{q_s} \lambda_j c_j^{(s)} \tag{26}$$

where

$$\lambda_\eta = \int_\Xi I(\boldsymbol{\xi}) \prod_{k \neq s} \sum_{j=0}^{q_k} \left[b_{j;q_k}(\tau_k(\xi_k)) \tilde{c}_j^{(k)} \right] b_{\eta;q_s}(\tau_s(\xi_s)) \mathrm{d}\boldsymbol{\xi} \tag{27}$$

This is a linear problem. Let $\tilde{j}$ be such that $\lambda_{\tilde{j}} = \min_j \lambda_j$. We can write our original problem as:

$$\min_{\mathbf{c}^{(s)}} \lambda_{\tilde{j}} + \sum_{j=0}^{q_s} (\lambda_j - \lambda_{\tilde{j}}) c_j^{(s)} \geq \lambda_{\tilde{j}} \tag{28}$$

It is now evident that $\min_{\mathbf{c}^{(s)}} \sum_{j=0}^{q_s} \lambda_j c_j^{(s)} = \lambda_{\tilde{j}}$, which can be obtained by making $c_{\tilde{j}}^{(s)} = 1$ and $c_j^{(s)} = 0 \; \forall j \neq \tilde{j}$. Altering $\tilde{\mathbf{c}}$ in this way does not change the value of E, since $\tilde{\mathbf{c}}$ minimized $\mathrm{E}(I, \mathbf{c})$, it must have also minimized Problem 26, and so we had $\sum_{j=0}^{q_s} \lambda_j \tilde{c}_j^{(s)} = \lambda_{\tilde{j}}$

By repeating this process for all s for which $c^{(s)}$ has more than one non-zero element, we obtain a new vector $\hat{\mathbf{c}}$ for which the condition in the lemma is satisfied, and for which $\mathrm{E}(I, \hat{\mathbf{c}}) = \mathrm{E}(I, \tilde{\mathbf{c}}) = \mathrm{E}_l(I)$.

□

These lemmas tell us that regardless of which family of pdfs, $\mathcal{P}_m$ or $\mathcal{P}_u$, we use in Program 9 to calculate E_l, the value we get could be obtained by using one of the multivariate Berstein basis distributions. Therefore, using either of those families is equivalent to using the following discrete set:

$$\mathcal{P}_j = \{ B_{\mathbf{j}} \left(\tau(\boldsymbol{\xi}) \right) \; \forall \mathbf{j} \in \mathcal{J} \subset \mathbb{N}^{n_\xi} \} \tag{29}$$

This allows us to reformulate our problem as the following integer problem:

$$\mathrm{E}_l(I) = \min_{\mathbf{j} \in \mathcal{J}} \mathrm{E}(I; \mathbf{j}) \tag{30}$$

where,

$$\mathrm{E}(I; \mathbf{j}) = \int_\Xi I(\boldsymbol{\xi}) \prod_k \mathcal{B}_{\mathbf{j}(k)}(\tau_k(\xi_k)) \mathrm{d}\boldsymbol{\xi} \tag{31}$$

This reduces our search space to just n_ξ dimensions, and will be the basis for the minimization algorithm we propose.

4.3 Minimization Algorithm

Since it is our requirement that the execution time of the algorithm is polynomial with the number of dimensions, going through each possible combination of indexes is out of question. Instead, a method similar to a pattern search is employed. This method, at each step, finds the mutation $j_k = x$ that most reduces the value of E, as in Equation 32. At each step there is a guarantee that the value of E is decreased. These steps are repeated until no improvements can be made.

$$\begin{aligned} \underset{\hat{k},x}{\operatorname{argmin}} \; & \mathrm{E}\left(I; \mathbf{j}^{(t+1)} \right) \\ & j_{\hat{k}}^{(t+1)} = x \\ & j_k^{(t+1)} = j_k^{(t)} \; \forall k \neq \hat{k} \end{aligned} \tag{32}$$

Note that when we calculate one metric y_i we also calculate all the other metrics, because all metrics require propagating the trajectory, which is where most of the execution time is spent. This means that when we calculate an expectation $\mathrm{E}(y_i < \nu_i; \mathbf{c})$ we also calculate the expectations for the other metrics. To reduce execution time, memoization is applied, i.e., when we evaluate $\mathrm{E}(y_i < \nu_i; \mathbf{c})$, we store $\mathrm{E}(y_j < \nu_j; \mathbf{c}) \; \forall j$ in case these values are needed in the estimations of $\mathrm{E}_l(y_j < \nu_j)$.

The process in Equation 32 always finds a local solution, but has no guarantee of finding the global optimum. The solution returned by this method depends on the initial guess $\mathbf{j}^{(0)}$.

One possible initial guess is to choose greedily each j_k, according to:

$$\forall k, \; j_k^{(0)G} = \underset{i}{\operatorname{argmin}} \int_\Xi I(\boldsymbol{\xi}) b_{i,q_k}(\xi_k) \mathrm{d}\boldsymbol{\xi} \tag{33}$$

The solution obtained using this initial guess will be termed $j^{(\omega)G}$. To improve the likelihood of finding the optimal solution, we re-start the optimization with a new initial guess. We want this initial guess to be unlikely to fall into the same local optimum as $j^{(\omega)G}$, so we define it, $\mathbf{j}^{(0)GO}$, as the complement of $\mathbf{j}^{(\omega)G}$, as defined in Equation

34. The solution obtained by applying the pattern search starting from $\mathbf{j}^{(0)GO}$ is compared with $\mathbf{j}^{(\omega)G}$, and the one which produces the lowest value of the expectation is kept and termed $\mathbf{j}^{(\omega)GO}$.

$$j_k^{(0)GO} = q_k - j_k^{(\omega)G} \tag{34}$$

Experimental Results

To assess the accuracy of this algorithm, an experiment was performed. For n_ξ ranging from 2 to 10, 1667 different values of w were sampled following the Halton sequence. For each of those, 3 indicator functions were created, one for each of the 3 metrics used in the test case in Section 6.1: propellant mass, distance and relative speed to target (Apophis). This produces 5001 different indicator functions, for each of which both versions of the algorithm were run. For n_ξ up to 5, the solution was obtained using a brute-force method, to obtain error metrics. For higher n_ξ, it was too time consuming to perform brute-force search. Table 2 shows the maximum absolute errors of the algorithm with and without the new starting point, and Table 3 shows the fraction of the errors that were below 10^{-6}.

Table 2: Maximum absolute error between E_l (obtained by brute-force search) and E obtained using either of the algorithms considered.

n_ξ	$j^{(\omega)G}$	$j^{(\omega)GO}$
2	0	0
3	0.0151	0
4	0.1046	0
5	0.1513	0.0001

Table 3: Percentage of absolute errors below 10^{-6}. Absolute error between E_l (obtained by brute-force search) and E obtained using either of the algorithms considered.

n_ξ	$j^{(\omega)G}$	$j^{(\omega)GO}$
2	100%	100%
3	99.8%	100%
4	97.3%	99.98%
5	97.5%	99.96%

In Figure 1 we can see the number of different values of $\mathbf{j}$ that are evaluated, on average, per indicator function. Note that this number has been reduced thanks to memoization as mentioned previously. For either of the algorithms, the number of function evaluations remains under 100 per indicator function.

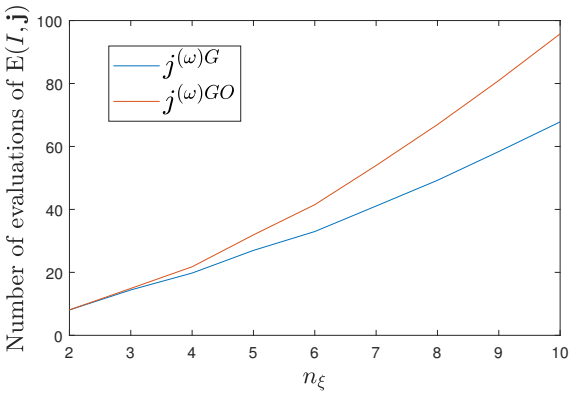

Fig. 1: Number of function evaluations per indicator function for which the lower expectation is to be calculated.

Given these results, we will be using $j^{(\omega)GO}$ to calculate E_l.

5. Multi Objective Optimization of Lower Expectation

Oftentimes the optimization of one objective conflicts with the optimization of another. It is not always possible to determine how to weigh each of them so as to obtain a unique solution. A solution is said to be dominated if there's a point in the search space that is better than it in every objective. Naturally, regardless of the importance the designers give to each of the objectives, they will always prefer to choose a non-dominated solution. Therefore, when we have a multi-objective problem, we wish to estimate the Pareto front, which is the set of all non-dominated solutions. A tool that obtains samples that approximate the Pareto front is called a multi-objective optimizer. We're using MACS for this purpose.

MACS is a population based memetic multi-objective optimisation algorithm [7]. It works by combining local search actions with social actions to move the agents towards the Pareto front. At the same time an archiving process is used to select which of the solutions found are kept. Only points which are not known to be dominated are kept in the archive. Even with this filter, the archive would grow unbounded, so in addition, the set of points that minimizes a metric similar to electrostatic potential is selected and the remaining are discarded.

5.1 Surrogate Models

In our work we employ Kriging models $\tilde{\mathrm{E}}_l$, using the DACE toolbox [8], to speed-up the optimization. These try to capture the relationship between w, ν and $\mathrm{E}_l(I(y_i < \nu_i))$. One model is trained to approximate the lower expectation for each metric:

$$\tilde{\mathrm{E}}_{l\ i}(w, \nu_i) \approx \mathrm{E}_l(I(y_i(U(w)) < \nu_i)) \tag{35}$$

At the start of our method, the proxy control variable w and the thresholds ν are sampled with 100 elements of the Halton sequence, and for each of those we calculate $\mathrm{E}_l(I(y_i(U(w)) < \nu_i)$. The Kriging model is then trained to fit the values of the lower expectation.

MACS is then called on these models, and the lower expectations for the solutions it returns are calculated accurately. These values are then introduced in the model, which is then re-trained, and this process is repeated several times. This part of the process is necessary because the model is imperfect and the optimizer might find points that are poorly represented in the model and which appear to be good solutions because of it. This approach also leads to a higher sampling rate where it matters most - near the Pareto front.

We use a Kriging model with linear regression functions, and exponential correlation functions. We found that the choice of the correlation model is critical to ensure the model converges.

Each time MACS is run, the points that it finds are in the Pareto set, which is a subset of the search space. Using Gaussians as correlation functions, when too many training points are added in a subset of the search space, the model diverges. The accuracy of the model at points distant from this subset worsen considerably. This does not happen with exponential correlation functions, which are more "local" in the sense that adding training points in one region of the set does not affect significantly the value of the model in distant points.

MACS is run with 10000 function evaluations and an archival size of 10 and population of 21 agents during 10 iterations of refining the model. Afterwards, it's run once on the final model with 252 agents and 100 elements in the archive with 100000 function evaluations. The archive at the end of running MACS contains the solutions, and in our case it was always full. All of these solutions are then used to refine the model.

6. Test Cases : Rendezvous with Apophis

We consider two test cases. In both of them we study a rendezvous mission with Apophis, but we consider two different formulations for uncertainty. In the first, we have uncertain engine parameters, and in the second we have uncertain engine outage.

Apophis is an asteroid that in 2004 was classified as level 4 in the Torino scale, the highest ever score, due to a probability of impact that reached 1 in 60 at its highest estimate [9]. Later information lead to this probability becoming practically 0, and the asteroid being reclassified as level 0 in Torino scale, indicating that it poses no threat. Nonetheless its orbit regularly intersects the Earth's, and on average it passes within two lunar orbits every five years [9].

This makes it an excellent test case for robust design of asteroid rendezvous missions, with the purpose of improving predictions about future approaches or even redirect it if ever necessary. For such missions it is important to guarantee that the objectives are met in the face of significant epistemic uncertainty.

We consider the robust optimization of a mission with a spacecraft equipped with an ion engine with uncertain engine parameters and Earth escape velocity. The orbital elements for Apophis, for the epoch 28 September 2008, were taken from the JPL small object database *.

The metrics y considered were the propellant mass spent m_p, the distance to target Δr and the relative speed to target Δv at rendezvous. We are running MACS to optimize the surrogate models of the lower expectations on these quantities. The thresholds for these quantities were also optimized simultaneously with the other quantities, as in Equation 2. Therefore, our optimization problems can be written as:

$$\min_{w,\nu} \begin{matrix} -\tilde{\mathrm{E}}_{l\ m_p}(w, \nu_{m_p}) \\ -\tilde{\mathrm{E}}_{l\ \Delta r}(w, \nu_{\Delta r}) \\ -\tilde{\mathrm{E}}_{l\ \Delta v}(w, \nu_{\Delta v}) \\ \nu_{m_p} \\ \nu_{\Delta r} \\ \nu_{\Delta v} \end{matrix} \tag{36}$$

The difference between the two test cases considered here is only on the definition of $\boldsymbol{\xi}$.

6.1 Uncertain Engine Parameters

We considered the engine thrust and Isp to vary with longitude in a way that is uncertain. This is modelled with uncertain variables T_i and $I_{sp,i}$ representing the value of these parameters at equispaced longitudes. The values at intermediate longitudes are obtained by linear interpolation. The Earth escape velocity v_∞ was also considered uncertain.

Therefore, variable $\boldsymbol{\xi}$ is given as

* https://ssd.jpl.nasa.gov/sbdb.cgi

Table 4: Lower and upper bounds that define Ξ, the hyper-rectangular space of uncertain variables, for the uncertain engine parameters test case.

	v_∞ [Km/s]	T_i [N]	$I_{sp,i}$ [s]
$\boldsymbol{\xi}^L$	3	2772	0.0477
$\boldsymbol{\xi}^U$	3.7	3388	0.0583

$$\boldsymbol{\xi} = [v_\infty \; T_1 \; \cdots \; T_{n_T} \; I_{sp,1} \; \cdots \; I_{sp,n_I}] \tag{37}$$

We considered $n_T = 5$ and $n_I = 4$ so that $n_\xi = 10$. This number was chosen to demonstrate the ability of our method to deal with high numbers of epistemic uncertain variables.

The uncertain variables are bounded according to Table 4, where $\boldsymbol{\xi}^L$ and $\boldsymbol{\xi}^U$ represent the lower and upper bounds, such that $\Xi = \left[\boldsymbol{\xi}^L \; \boldsymbol{\xi}^U\right]$.

Results

In Table 5 are shown a selection of solutions found by this method. To visualize the nature of the trade-offs, a Pareto front with fixed thresholds was obtained, which is shown in Figure 2.

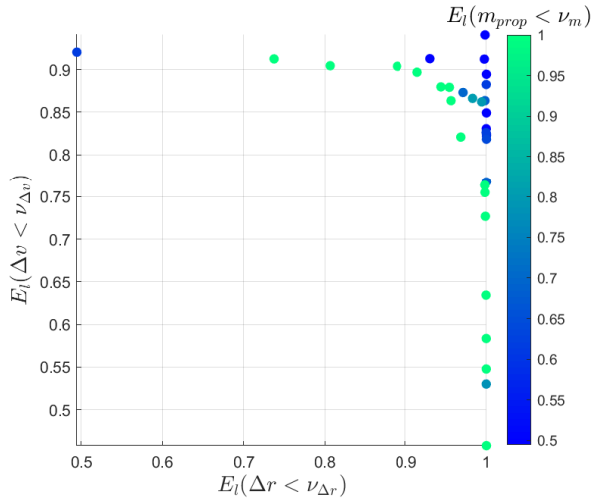

Fig. 2: Scatter plot of Pareto front for lower expectations with fixed thresholds. The values of the thresholds ν_{m_p}, $\nu_{\Delta r}$ and $\nu_{\Delta v}$ are 45Kg, 0.018AU and 0.7Km/s, respectively. The colors indicate $\mathrm{E}_l\left(m_{prop} < \nu_{m_{prop}}\right)$.

As an illustration, the seventh row solution from Table 5 was plotted for 200 different $\boldsymbol{\xi}$ values to produce a line whose thickness can serve as a visualization of the uncertainty in position (Figure 3).

Table 5: A selection of solutions returned by our method. Solutions were selected using the same archival algorithm employed by MACS, after solutions with $\mathrm{E}_l \leq 0.01$ were removed.

E_{lm}	$\mathrm{E}_{l\Delta r}$	$\mathrm{E}_{l\Delta v}$	ν_m [Kg]	$\nu_{\Delta r}$ [AU]	$\nu_{\Delta v}$ [Km/s]
0.690	1.000	1.000	50.797	0.041	2.099
1.000	1.000	1.000	52.169	0.042	2.155
1.000	0.062	1.000	79.476	0.023	3.655
1.000	1.000	0.153	80.517	0.066	0.764
1.000	0.698	0.581	80.605	0.021	0.792
1.000	0.447	1.000	52.282	0.026	0.939
1.000	1.000	0.745	52.070	0.067	0.673
0.848	0.797	1.000	45.168	0.024	3.627
1.000	0.309	1.000	52.178	0.027	3.614
1.000	1.000	0.921	81.000	0.029	0.943

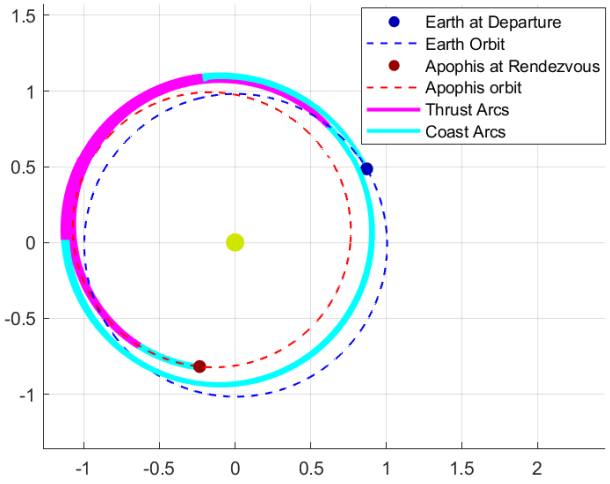

Fig. 3: Plot of trajectories for the control law corresponding to the last solution in Table 5 and for 200 different samples of $\boldsymbol{\xi}$, so that the thickness can serve to visualize the uncertainty in position. The plot is seen perpendicularly to the ecliptic plane and the axis are in AU.

6.2 Engine Outage

A different type of unexpected engine behaviour is an engine outage. In our case we considered as random variables the location of the beginning of the outage L_O, its duration ΔL_O, and intensity η_O. For longitudes between L_O and $L_O + \Delta L_O$, the available engine thrust, which would be T

Table 6: Lower and upper bounds that define Ξ, the hyper-rectangular space of uncertain variables, for the engine outage test case.

	v_∞ [Km/s]	L_O [rad]	ΔL_O [rad]	η_O
$\boldsymbol{\xi}^L$	3	0	0	0
$\boldsymbol{\xi}^U$	3.7	4π	$\pi/8$	1.2

without the outage, becomes $\eta_O T$. We still consider v_∞ as an uncertain variable.

The uncertain vector $\boldsymbol{\xi}$ for this test case is given as

$$\boldsymbol{\xi} = [v_\infty \; L_O \; \Delta L_O \; \eta_O] \tag{38}$$

As an example of engine outage being applied to a control vector consider the case illustrated in Figure 4. The curves represents the engine thrust as a fraction of its maximum value, with and without the effect of outage.

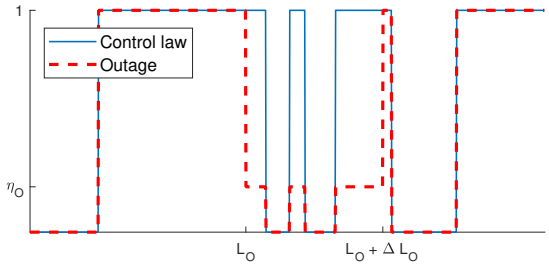

Fig. 4: Example of the effect of engine outage as modelled in this work. Both lines represent the fraction of maximum available thrust as a function of the longitude. The blue solid line represents this fraction as specified by the control vector, and the red dashed line is the thrust that is actually applied due to the outage.

The bounds for $\boldsymbol{\xi}$ for this test case are represented in Table 6.

Results

Once again we obtain a table with a selection of solutions returned by our method, in Table 7, and a plot for many values of $\boldsymbol{\xi} \in \Xi$ in Figure 5. Running the optimization with fixed thresholds produces a 3D plot as the one in Figure 6.

Table 7: A selection of solutions returned by our method. Solutions were selected using the same archival algorithm employed by MACS, after solutions with $\mathrm{E}_l \leq 0.01$ were removed.

E_{lm}	$\mathrm{E}_{l\Delta r}$	$\mathrm{E}_{l\Delta v}$	ν_m [Kg]	$\nu_{\Delta r}$ [AU]	$\nu_{\Delta v}$ [Km/s]
0.113	0.431	0.432	33.968	0.023	1.205
0.121	0.383	0.385	36.581	0.025	1.298
1.000	1.000	1.000	44.588	0.009	3.524
1.000	1.000	0.089	60.265	0.020	0.283
0.110	1.000	1.000	28.911	0.065	2.968
1.000	0.328	1.000	76.775	0.009	3.598
1.000	0.269	0.261	44.370	0.064	0.692
0.262	0.058	1.000	41.975	0.007	3.432
0.290	1.000	0.137	41.891	0.018	0.558
1.000	1.000	0.548	70.717	0.027	0.653

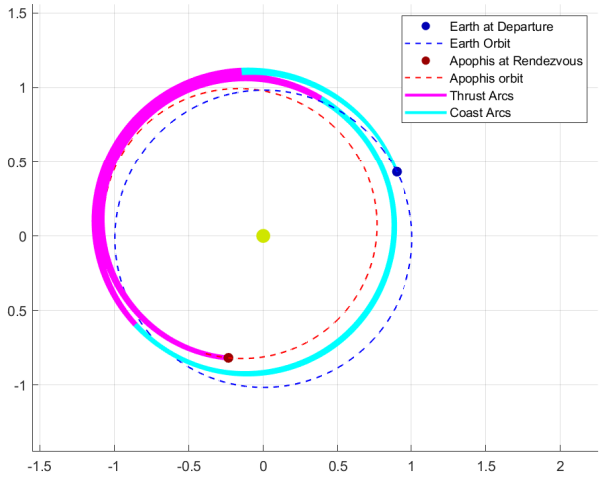

Fig. 5: Plot of trajectories for the control law corresponding to the last solution in Table 7 and for 200 different samples of $\boldsymbol{\xi}$, so that the thickness can serve to visualize the uncertainty in position. The plot is seen perpendicularly to the ecliptic plane and the axis are in AU.

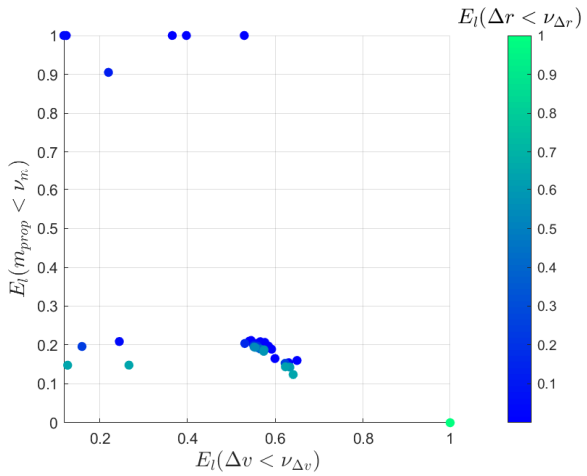

Fig. 6: Scatter plot of Pareto front for lower expectations with fixed thresholds. The values of the thresholds ν_{m_p}, $\nu_{\Delta r}$ and $\nu_{\Delta v}$ are 35Kg, 0.005AU and 0.75Km/s, respectively. The colors indicate $\mathrm{E}_l\,(\Delta r < \nu_{\Delta r})$.

6.3 Execution Times

The execution time can be split into three parts, the computation of E_l and the control map, to create the training points for the surrogate model, and running MACS on the surrogate model. The following times were obtained in Matlab(R) 2018b on a Windows 10 computer with Intel(R) Core(TM) i7-8700 @3.2GHz and 8GB of RAM.

- Running MACS on the surrogate model takes about 10 seconds per iteration with 10000 function evaluations, and 40 seconds for the final iteration with 100000.
- The control map (Problem 8) takes about 3.7 seconds, to calculate u from w.
- Solving Problem 30 to obtain E_l from u took 4.5 seconds for the first test case (variable thrust and Isp) and 7.7 seconds for the second (engine outage). The longer time with engine outage is because the calculation of the values of the indicator function for the quasi-Monte-Carlo estimation of $\mathrm{E}(I;\mathbf{c})$ could not be fully vectorized.

There are in total 300 points for which we calculate E_l, 100 for the initial surrogate, 10 per iteration and 100 for the final iteration. In total, the first test case took 45 minutes and the second 1 hour, approximately.

7. Conclusions

This paper has presented a method for obtaining robust solutions for the optimal control problem of finding a trajectory to rendezvous with an asteroid. The uncertainty was modelled using lower expectation, a method which allows taking into account epistemic uncertainty. The family of distributions used to define the lower expectation was based on Bernstein polynomials.

A method to solve the optimisation problem required to calculate lower expectation without exponential complexity on the number of uncertain variables was also developed.

We applied our method to two test cases, both of which consider a rendezvous mission with asteroid Apophis. The first considered an engine with unknown parameters, which evolve over time in an unknown way. The second applied this process to an engine outage, where for a small amount of time, the engine's thrust suddenly changes in an unpredictable way.

This method could be applied to any design problem subject to epistemic uncertainty with multiple possibly conflicting objectives. As future work we will apply this method to a trajectory with more than one target, such as an asteroid tour.

Acknowledgement

This research has been developed with the partial support of the CNES grant R-S17/BS-0005-034 "Robust Optimization of Low-Thrust Interplanetary Trajectories" and the H2020 MCSA ITN UTOPIAE grant agreement number 722734.

References

[1] Marilena Di Carlo and Massimiliano Vasile. "Robust Optimisation of Low-Thrust Interplanetary Transfers Using Evidence Theory". In: AAS/AIAA Space Flight Mechanics Meeting. Jan. 13, 2019.

[2] Marilena Di Carlo, Juan Manuel Romero Martin, and Massimiliano Vasile. "CAMELOT: Computational-Analytical Multi-fidElity Low-thrust Optimisation Toolbox". In: *CEAS Space Journal* 10.1 (Mar. 1, 2018), pp. 25–36. ISSN: 1868-2510. DOI: 10.1007/s12567-017-0172-6. URL: https://doi.org/10.1007/s12567-017-0172-6 (visited on 01/25/2019).

[3] Federico Zuiani and Massimiliano Vasile. "Extended analytical formulas for the perturbed Keplerian motion under a constant control acceleration". In: *Celestial Mechanics and Dynamical Astronomy* 121.3 (Mar. 1, 2015), pp. 275–300. ISSN: 1572-9478. DOI: 10.1007/s10569-014-9600-5. URL: https:

//doi.org/10.1007/s10569-014-9600-5 (visited on 04/12/2019).

[4] Massimiliano Vasile and Chiara Tardioli. "On the Use of Positive Polynomials for the Estimation of Upper and Lower Expectations in Orbital Dynamics". In: *Stardust Final Conference* (2018), pp. 99–107. DOI: 10.1007/978-3-319-69956-1_6. URL: https://link.springer.com/chapter/10.1007/978-3-319-69956-1_6 (visited on 03/05/2019).

[5] Clemens Heitzinger. "Simulation and Inverse Modeling of Semiconductor Manufacturing Processes". dissertation. Technischen Universität Wien, Nov. 2002.

[6] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. en. CBMS-NSF regional conference series in applied mathematics 63. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 1992. ISBN: 978-0-89871-295-7.

[7] Lorenzo A. Ricciardi and Massimiliano Vasile. "Improved Archiving and Search Strategies for Multi Agent Collaborative Search". In: *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*. Ed. by Edmondo Minisci et al. Computational Methods in Applied Sciences. Cham: Springer International Publishing, 2019, pp. 435–455. ISBN: 978-3-319-89988-6. DOI: 10.1007/978-3-319-89988-6_26. URL: https://doi.org/10.1007/978-3-319-89988-6_26 (visited on 01/25/2019).

[8] H. B. Nielsen, S. N. Lophaven, and J. Søndergaard. *DACE - A Matlab Kriging Toolbox*. Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002. URL: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=1460.

[9] Don Yeomans, Steve Chesley, and Paul Chodas. *Near-Earth Asteroid 2004 MN4 Reaches Highest Score To Date On Hazard Scale*. 2004. URL: https://cneos.jpl.nasa.gov/news/news146.html (visited on 09/17/2019).